CLAIMS

1. A method for performing credential and condition assertion verification corresponding to a policy file, comprising:

during an initialization process, dynamically creating comparing functions for principals, said principals having credentials, said credentials from said policy file, and dynamically creating comparing functions for states of protocol events, said events having conditions, said conditions from said policy file;

during said initialization process, dynamically creating and loading a module, said module containing said comparing functions;

during runtime, ensuring an installed policy file corresponds to said module, and, if not, repeating said initialization process using said installed policy file, thereby dynamically generating an updated module containing updated comparing functions, said updated module and said updated comparing functions corresponding to said installed policy file; and

calling said comparing functions as appropriate.

2. The method of Claim 1, further comprising:

using in a high level language to generate said module.

3. A method for performing credential and condition assertion verification corresponding to a policy file, said policy file comprising credentials, conditions, and a hash value, said method comprising:

loading said policy file into an in-memory representation;

requesting loading an assertion verification dynamically loadable library, herein referred to as DLL, said DLL comprising a predetermined hash return function, principal/credential comparing functions, and protocol/condition comparing functions;

if said DLL exists:

loading said DLL into said memory; and

calling a predetermined function in said DLL for a return value, whereby said loading is complete if said returned value equals said hash value of said policy file;

if said DLL does not exist or if said loading said DLL is not complete:

invoking a code generation function for generating an updated assertion verification DLL from an assertion code file, said generated DLL corresponding to said policy file;

compiling and linking said assertion code file, thereby generating said updated assertion verification DLL corresponding to said policy file;

loading said updated assertion verification DLL into said memory; and

during runtime, calling said comparing functions in said DLL in memory as appropriate.

4. The method of Claim 3, further comprising said code generation function:

adding header information to said assertion code file;

adding a predetermined function that returns said hash value of said policy file;

interating through said credentials of said loaded policy file for generating said principal/credential comparing functions; and

interating through said conditions of said loaded policy file for generating said protocol/condition comparing functions.

5. The method of Claim 4, wherein said principal/credential comparing functions perform:

calling other credential comparison methods for any credentials used in definition of said each credential;

making calls to other comparison operations based on allowable operations of built-in types of a policy language corresponding to said policy file; and

combining results of above comparisons using logical operators.

6. The method of Claim 4, wherein said protocol/condition comparing functions perform:

calling other condition comparison methods for any conditions used in definition of said each condition;

making calls to other comparison operations based on allowable operations of built-in types of a policy language corresponding to said policy file; and

combining results of above comparisons using logical operators.

7. The method of Claim 3, during runtime, further comprising:

each time for deciding if a principal is described by a tested credential, computing a name of a comparison function of said principal/credential

comparing functions, said comparison function name based on name of said tested credential;

calling said principal/credential comparison function, said principal/credential comparison function returning a value representing if said tested credential matches said principal;

each time for deciding if a protocol state satisfies a tested condition, computing a name of a comparison function of said protocol/condition comparing functions, said comparison function name based on name of said tested condition; and

calling said protocol/condition comparison function, said protocol/condition comparison function returning a value representing if said tested condition matches said protocol state.

8. An apparatus for performing credential and condition assertion verification corresponding to a policy file, said policy file comprising credentials, conditions, and a hash value, said apparatus comprising:

means for loading said policy file into an in-memory representation;

means for requesting loading an assertion verification dynamically loadable library, herein referred to as DLL, said DLL comprising a predetermined hash return function, principal/credential comparing functions, and protocol/condition comparing functions;

if said DLL exists:

means for loading said DLL into said memory; and

means for calling a predetermined function in said DLL for a return value, whereby said loading is complete if said returned value equals said hash value of said policy file;

if said DLL does not exist or if said loading said DLL is not complete:

means for invoking a code generation function for generating an updated assertion verification DLL from an assertion code file, said generated DLL corresponding to said policy file;

means for compiling and linking said assertion code file, thereby generating said updated assertion verification DLL corresponding to said policy file;

means for loading said updated assertion verification DLL into said memory; and

during runtime, means for calling said comparing functions in said DLL in memory as appropriate.

9. The apparatus of Claim 8, said code generation function further comprising:

means for adding header information to said assertion code file;

means for adding a predetermined function that returns said hash value of said policy file; and

means for interating through said credentials of said loaded policy file for generating said principal/credential comparing functions; and

means for interating through said conditions of said loaded policy file for generating said protocol/condition comparing functions.

10. The apparatus of Claim 9, said principal/credential comparing functions further comprising:

means for calling other credential comparison methods for any credentials used in definition of said each credential;

means for making calls to other comparison operations based on allowable operations of built-in types of a policy language corresponding to said policy file; and

means for combining results of above comparisons using logical operators.

11. The apparatus of Claim 9, said protocol/condition comparing functions further comprising:

means for calling other condition comparison methods for any conditions used in definition of said each condition;

means for making calls to other comparison operations based on allowable operations of built-in types of a policy language corresponding to said policy file; and

means for combining results of above comparisons using logical operators.

12. The apparatus of Claim 8, during runtime, further comprising:

each time for deciding if a principal is described by a tested credential, means for computing a name of a comparison function of said principal/credential comparing functions, said comparison function name based on name of said tested credential;

means for calling said principal/credential comparison function, said principal/credential comparison function returning a value representing if said tested credential matches said principal;

each time for deciding if a protocol state satisfies a tested condition, means for computing a name of a comparison function of said

protocol/condition comparing functions, said comparison function name based on name of said tested condition; and

means for calling said protocol/condition comparison function, said protocol/condition comparison function returning a value representing if said tested condition matches said protocol state.